```
1
```

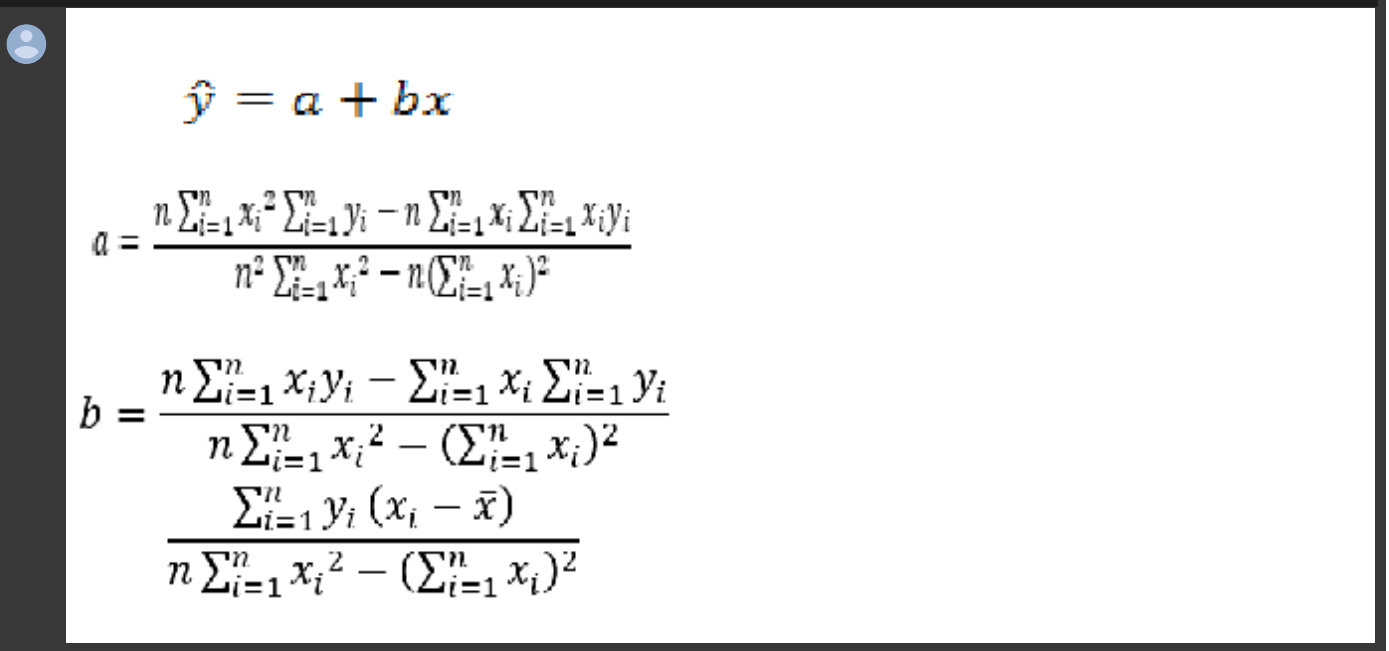
Simple Linear Regression

Simple linear regression is a regression model that estimates the relationship between one independent variable and one dependent variable using a straight line. Both variables should be quantitative.

For example, the relationship between temperature and the expansion of mercury in a thermometer can be modeled using a straight line: as temperature increases, the mercury expands. This linear relationship is so certain that we can use mercury thermometers to measure temperature.

```
from PIL import Image
from IPython.core.display import Image, display
display(Image("Regression.png", width=300, height=300))

#Underlying equation of simple linear regression.
```



$$\hat{y} = a + bx$$

$$a = \frac{n\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - n\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n^2 \sum_{i=1}^{n} x_i^2 - n(\sum_{i=1}^{n} x_i)^2}$$

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$\frac{\sum_{i=1}^{n} y_i (x_i - \bar{x})}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

Create a variable "random_variable" of size 100 with variance = 5.

```
import numpy as np
np.random.seed(0)
random_variable = np.random.normal(0,5**0.5,100)
#here the first argument is mean and
#the second argument is standard deviation
print(random_variable) # if variance is 5 then standard deviation is 5**0.5.
print("Variance of the sample:",np.var(random_variable))
```

```
[ 3.94454096  0.89477872  2.18852466  5.01078952  4.17598662 -2.18525977
  2.12446229 -0.33844501 -0.23080437  0.91812616  0.32209122  3.25185442
  1.70173209  0.27207361  0.99250836  0.74611848  3.34086237 -0.45874782
  0.70004066 -1.90981613 -5.70865877  1.46153561  1.9329381  -1.65953144
  5.07532563 -3.25206051  0.10231916 -0.41855581  3.42739852  3.28558609
  0.34647298  0.8455971  -1.98514928 -4.42919555 -0.77795522  0.34960692
  2.75101359  2.68860308 -0.86608909 -0.6759695  -2.34463571 -3.17525664
 -3.81533613  4.36206639 -1.13961692 -0.97956392 -2.80133559  1.73852129
 -3.6087853  -0.47570173 -2.0023241   0.86514029 -1.14219501 -2.63997382
 -0.06301738  0.95777918  0.14873703  0.67634772 -1.41838732 -0.81111391
 -1.50366727 -0.80398531 -1.81825036 -3.86008525  0.39673692 -0.89840949
 -3.64523432  1.03481258 -2.02878082  0.11615344  1.63029606  0.28841456
  2.54777738 -2.76115447  0.89966326 -1.53128192 -1.94716162 -1.2943472
 -0.69665264  0.12558952 -2.60535425  2.01430926  1.04125287 -3.43514531
  3.32783307  4.23933708  2.63583125 -0.40232416 -2.39427565  2.35782574
 -0.90153106  2.73347028  0.46571701  2.18383128  0.79685949  1.57994564
  0.02347876  3.99332782  0.28278407  0.89887554]
```

Create a random variable sequence("y") of size 100,

where y = random_sequence + 2*x + 1

```
1 np.random.seed(0)
2 #again we use the random seed generator
3
4 y = np.random.randn(100) + 2*random_variable + 1
5 print(y)
```

```
[ 10.65313427   3.18971465   6.35578731  13.26247225  11.21953123
  -4.34779742   6.19901299   0.17175278   0.43517241   3.24685083
   1.788226     8.95798235   5.1645019    1.66582223   3.42887996
   2.82591128   9.17580382  -0.12265391   2.71314903  -3.673728
 -12.97030736   4.57668982   5.7303124   -3.06122789  13.42040589
  -6.9584867    1.25039683  -0.02429548   9.38757625   9.04053096
   1.84789338   3.06935672  -3.85808431  -9.83918757  -0.90382258
   1.85556282   7.73231787   7.579586    -1.119505    -0.65424175
  -4.73782438  -6.77053121  -8.33694246  11.67490818  -1.78888603
  -1.39720214  -5.85546653   5.25453293  -7.83146844  -0.16414374
  -3.90011477   3.11718307  -1.79519516  -5.46057982   0.84578302
   3.34389023   1.36399128   2.65516735  -2.47109674  -0.98496898
  -2.67979499  -0.96752378  -3.44964701  -8.4464531    1.97089997
  -1.19859991  -7.92066699   3.53240742  -3.96486      1.28425227
   4.98968268   1.70581202   7.23495545  -5.75713477   3.20166816
  -2.74737392  -3.76512039  -2.16754406  -0.70485781   1.30734439
  -5.37585834   5.92944501   3.54816818  -7.40653431   9.14391834
  11.37456333   7.45044207   0.01542684  -4.85930392   6.77010321
  -1.20623907   7.68938562   2.139709     6.34430159   2.95008538
   4.86646444   1.05745754  10.77252614   1.69448023   3.19974045]
```

Calculate a & b for the equation y = a+bx

```
1 from scipy.stats import linregress
2 #scipy.stats.linregress(x, y=None)[source]
3 #used to Calculate a linear least-squares regression for two sets of measurements.
4 #Parameters like here we have taken are two variables
5 #x, y
```

```
#array_like
#Two sets of measurements. Both arrays should have the same length. If only x is given
#two-dimensional array where one dimension has length 2. The two sets of measurements a
#array along the length-2 dimension. In the case where y=None and x is a 2x2 array, lin
lg = linregress(random_variable,y)
a = lg[0]
b = lg[1]
print("Slope:",a)
print("Intercept:",b)
```

```
Slope: 2.447213595499957
Intercept: 1.0
```

Calculate yhat,

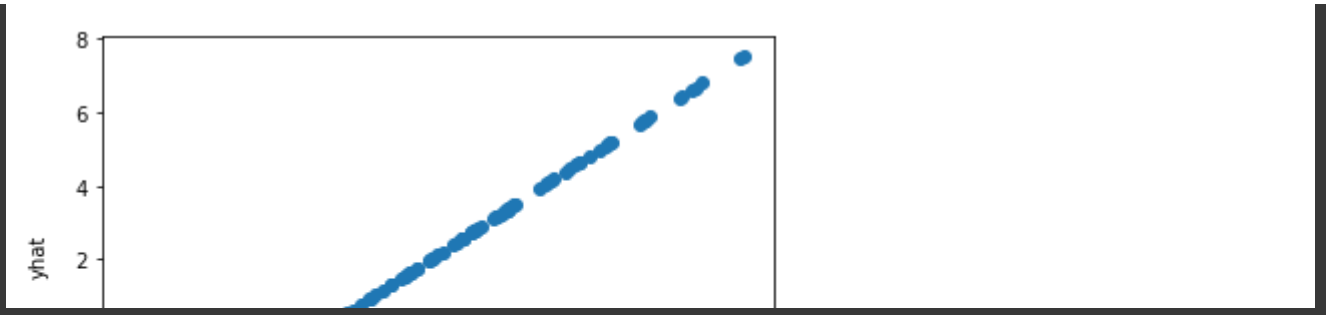
yhat = a+bx

```
yhat = a + b*random_variable
print(yhat)
```

```
[ 6.39175456  3.34199232  4.63573826  7.45800312  6.62320021  0.26195382
  4.57167588  2.10876859  2.21640923  3.36533976  2.76930481  5.69906801
  4.14894568  2.7192872   3.43972196  3.19333207  5.78807597  1.98846577
  3.14725426  0.53739746 -3.26144518  3.90874921  4.3801517   0.78768216
  7.52253923 -0.80484692  2.54953275  2.02865778  5.87461211  5.73279969
  2.79368657  3.2928107   0.46206431 -1.98198196  1.66925838  2.79682052
  5.19822719  5.13581667  1.5811245   1.7712441   0.10257789 -0.72804304
 -1.36812254  6.80927999  1.30759667  1.46764968 -0.35412199  4.18573488
 -1.1615717   1.97151187  0.44488949  3.31235388  1.30501858 -0.19276022
  2.38419622  3.40499277  2.59595063  3.12356132  1.02882627  1.63609969
  0.94354632  1.64322828  0.62896323 -1.41287165  2.84395051  1.54880411
 -1.19802073  3.48202618  0.41843278  2.56336703  4.07750965  2.73562815
  4.99499098 -0.31394088  3.34687686  0.91593168  0.50005198  1.1528664
  1.75056096  2.57280312 -0.15814065  4.46152286  3.48846647 -0.98793172
  5.77504667  6.68655067  5.08304485  2.04488943  0.05293795  4.80503934
  1.54568254  5.18068387  2.9129306   4.63104487  3.24407308  4.02715923
  2.47069236  6.44054142  2.73099766  3.34608914]
```

Scatterplot x and yhat

```
import matplotlib.pyplot as plt
plt.scatter(random_variable,yhat)
plt.xlabel("x")
plt.ylabel("yhat")
plt.show()
```



Determine how good the model is by computing the r-squared

https://www.khanacademy.org/math/ap-statistics/bivariate-data-ap/assessing-fit-least-squares-regression/a/r-squared-intuition

Calculate the goodness of fit for the above model(R-squared).

```
# This is formatted as code
```

```
1 # to calculate R-squared
2 r_sq = lg[2]
3 print("R-squared :",r_sq)
```

```
R-squared : 0.9999999999999996
```